

Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks

Hao Zhuang, *Student Member, IEEE*, Wenjian Yu, *Senior Member, IEEE*, Shih-Hung Weng, Ilgweon Kang, *Student Member, IEEE*, Jeng-Hau Lin, *Student Member, IEEE*, Xiang Zhang, *Member, IEEE*, Ryan Coutts, and Chung-Kuan Cheng, *Fellow, IEEE*

Abstract—We design an algorithmic framework using matrix exponentials for time-domain simulation of power delivery network (PDN). Our framework can reuse factorized matrices to simulate the large-scale linear PDN system with variable step-sizes. In contrast, current conventional PDN simulation solvers have to use fixed step-size approach in order to reuse factorized matrices generated by the expensive matrix decomposition. Based on the proposed exponential integration framework, we design a PDN solver *R-MATEX* with the flexible time-stepping capability. The key operation of matrix exponential and vector product (MEVP) is computed by the rational Krylov subspace method.

To further improve the runtime, we also propose a distributed computing framework *DR-MATEX*. *DR-MATEX* reduces Krylov subspace generations caused by frequent breakpoints from a large number of current sources during simulation. By virtue of the superposition property of linear system and scaling invariance property of Krylov subspace, *DR-MATEX* can divide the whole simulation task into subtasks based on the alignments of breakpoints among those sources. The subtasks are processed in parallel at different computing nodes without any communication during the computation of transient simulation. The final result is obtained by summing up the partial results among all the computing nodes after they finish the assigned subtasks. Therefore, our computation model belongs to the category known as *Embarrassingly Parallel* model.

Experimental results show *R-MATEX* and *DR-MATEX* can achieve up to around $14.4\times$ and $98.0\times$ runtime speedups over traditional trapezoidal integration based solver with fixed time-step approach.

Index Terms—Circuit simulation, power delivery/distribution networks, power grid, time-domain simulation, transient simulation, matrix exponential, Krylov subspace method, parallel processing.

Manuscript received in May 2015; Revised in August, 2015 and November 2015. Accepted in Jan, 2016.

This work was supported in part by nsf-ccf 1017864, NSFC 61422402 and NSFC 61274033.

H. Zhuang, I. Kang, J.-H. Lin, and C. K. Cheng are with the Department of Computer Science and Engineering, University of California, San Diego, CA. (e-mail: hao.zhuang@cs.ucsd.edu, igkang@ucsd.edu, jel252@eng.ucsd.edu, ckcheng@ucsd.edu)

W. Yu is with Tsinghua National Laboratory for Information Science and Technology, the Department of Computer Science and Technology, Tsinghua University, Beijing, China. (e-mail: yu-wj@tsinghua.edu.cn)

X. Zhang and R. Coutts are with Department of Electrical and Computer Engineering, University of California, San Diego, CA. (e-mail: xiz110@eng.ucsd.edu, rmcoutts@eng.ucsd.edu)

S.-H. Weng was with the Department of Computer Science and Engineering, University of California, San Diego, CA. He is now with Facebook, Inc., Menlo Park, CA, USA. (e-mail: shweng@fb.com)

I. INTRODUCTION

MODERN VLSI design verification relies heavily on the analysis of power delivery network (PDN) to estimate power supply noises [1]–[8]. The performance of power delivery network highly impacts on the quality of global, detailed and mixed-size placement [9]–[11], clock tree synthesis [12], global and detailed routing [13], as well as timing [14] and power optimization. Lowering supply voltages, increasing current densities as well as tight design margins demand more accurate large-scale PDN simulation. Advanced technologies [15], [16], three dimensional (3D) IC structures [17]–[19], and increasing complexities of system designs all make VLSI PDNs extremely huge and the simulation tasks time-consuming and computationally challenging. Due to the enormous size of modern designs and long simulation runtime of many cycles, instead of general nonlinear circuit simulation [20], [21], PDN is often modeled as a large-scale linear circuit with voltage supplies and time-varying current sources [22]–[24]. Those linear matrices are obtained by parasitic extraction process [4], [25]–[28]. After those processes, we need time-domain large-scale linear circuit simulation to obtain the transient behavior of PDN with above inputs.

Traditional methods in linear circuit simulation solve differential algebra equations (DAE) numerically in explicit ways, e.g., forward Euler (FE), or implicit ways, e.g., backward Euler (BE) and trapezoidal (TR), which are all based on low order polynomial approximations for DAEs [29]. Due to the stiffness of systems, which comes from a wide range of time constants of a circuit, the explicit methods require extremely small time step sizes to ensure the stability. In contrast, implicit methods can handle this problem with relatively large time steps because of their larger stability regions. However, at each time step, these methods have to solve a linear system, which is sparse and often ill-conditioned. Due to the requirement of a robust solution, compared to iterative methods [30], direct methods [31] are often favored for VLSI circuit simulation, and thus adopted by state-of-the-art power grid (PG) solvers in TAU PG simulation contest [32]–[34]. Those solvers only require one matrix factorization (LU or Cholesky factorization) at the beginning of the transient simulation. Then, at each fixed time step, the following transient computation requires only pairs of forward and backward substitutions, which achieves better efficiency over adaptive stepping methods by reusing the

factorization matrix [24], [32], [34] in their implicit numerical integration framework. However, the maximum of step size choice is limited by the smallest distance h_{upper} among the breakpoints [35]. Some engineering efforts are spent to break this limitation by sacrificing the accuracy. In our work, we always obey the upper limit h_{upper} of time step to maintain the fidelity of model, which means the fixed time step h cannot go beyond h_{upper} in case of missing breakpoints.

Beyond traditional methods, a class of methods called matrix exponential time integration has been embraced by MEXP [36]. The major complexity is caused by matrix exponential computations. MEXP utilizes standard Krylov subspace method [37] to approximate matrix exponential and vector product. MEXP can solve the DAEs with much higher order polynomial approximations than traditional ones [36], [37]. Nevertheless, when simulating stiff circuits with standard Krylov subspace method, it requires the large dimension of subspace in order to preserve the accuracy of MEXP approximation and poses memory bottleneck and degrade the adaptive stepping performance of MEXP.

Nowadays, the emerging multi-core and many-core platforms bring powerful computing resources and opportunities for parallel computing. Even more, cloud computing techniques [38] drive distributed systems scaling to thousands of computing nodes [39]–[41], etc. Distributed computing systems have been incorporated into products of many leading EDA companies and in-house simulators [42]–[46]. However, building scalable and efficient distributed algorithmic framework for transient linear circuit simulation is still a challenge to leverage these powerful computing tools. The papers [47], [48] show great potentials by parallelizing matrix exponential based method to achieve the runtime performance improvement and maintain high accuracy.

In this work, we develop a transient simulation framework using matrix exponential integration scheme, *MATEX*, for PDN simulation. Following are the challenges we need to address. First, when the circuit is stiff, the standard Krylov subspace has convergence problem and slows down the computation of MEVP. Second, the frequent time breakpoints due to the transitions of PDN current sources modeling triggers the generations of Krylov subspace. Therefore, we might gain performance where we leverage the large time stepping, but we also lose runtime for the small step size. Our contributions are listed as below:

- MEVP in *MATEX* is efficiently computed by rational or invert Krylov subspace method. Compared to the commonly adopted framework using TR with fixed time step (TR-FTS), the proposed *MATEX* can reuse factorized matrix at the beginning of transient simulation to perform flexible adaptive time stepping.
- Among different Krylov subspace methods, we find rational Krylov subspace is the best strategy for MEVP in PDN simulation. Therefore, we design R-*MATEX* based on that and achieve up to around $15\times$ runtime speedup against the benchmarks over the traditional method TR-FTS with good accuracy.
- Furthermore, DR-*MATEX* is designed to improve R-*MATEX* with distributed computing resources.

- First, PDN's current sources are partitioned into groups based on their alignments. They are assigned to different computing nodes. Each node runs its corresponding PDN transient simulation task and has no communication overhead with other nodes.
- After all nodes finish the simulation computations, the results are summed up based on the linear superposition property of the PDN system.
- Proposed current source partition can reduce the chances of generating Krylov subspaces and prolong the time periods of reusing computed subspace at each node, which brings huge computational advantage and achieves up to $98\times$ speedup over traditional method TR-FTS.

The rest of this paper is organized as follows. Sec. II introduces the background of linear circuit simulation and matrix exponential formulations. Sec. III illustrates the Krylov techniques to accelerate matrix exponential and vector product computation. Sec. IV presents *MATEX* circuit solver and the parallel framework DR-*MATEX*. Sec. V shows numerical results and Sec. VI concludes this paper.

II. BACKGROUND

A. Transient Simulation of Linear Circuit

Transient simulation of linear circuit is the foundation of modern PDN simulation. It is formulated as DAEs via modified nodal analysis (MNA),

$$\mathbf{C}\dot{\mathbf{x}}(t) = -\mathbf{G}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (1)$$

where \mathbf{C} is the matrix for capacitive and inductive elements. \mathbf{G} is the matrix for conductance and resistance, and \mathbf{B} is the input selector matrix. $\mathbf{x}(t)$ is the vector of time-varying node voltages and branch currents. $\mathbf{u}(t)$ is the vector of supply voltage and current sources. In PDN, such current sources are often characterized as pulse or piecewise-linear inputs [22], [24] to represent the activities under the networks. To solve Eq. (1) numerically, the system is discretized with time step h and transformed to a linear algebraic system. Given an initial condition $\mathbf{x}(0)$ from DC analysis or previous time step $\mathbf{x}(t)$ and a time step h , $\mathbf{x}(t+h)$ can be obtained by traditional *low order approximation* methods [29].

B. Traditional Low Order Time Integration Schemes

1) *BE*: Backward Euler based time integration scheme (Eq.(2)) is a robust implicit first-order method.

$$\left(\frac{\mathbf{C}}{h} + \mathbf{G}\right)\mathbf{x}(t+h) = \frac{\mathbf{C}}{h}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t+h). \quad (2)$$

2) *TR*: Trapezoidal based time integration scheme (Eq.(3)) is a popular implicit second-order method.

$$\begin{aligned} \left(\frac{\mathbf{C}}{h} + \frac{\mathbf{G}}{2}\right)\mathbf{x}(t+h) &= \left(\frac{\mathbf{C}}{h} - \frac{\mathbf{G}}{2}\right)\mathbf{x}(t) \\ &+ \mathbf{B}\frac{\mathbf{u}(t) + \mathbf{u}(t+h)}{2}. \end{aligned} \quad (3)$$

It is probably the most commonly used strategy for large-scale circuit simulation, which has higher accuracy than BE.

3) *BE-FTS and TR-FTS*: Methods BE and TR with fixed time step (FTS) h are efficient approaches, which were adopted by the top PG solvers in 2012 TAU PG simulation contest [24], [32]–[34]. If only one h is used for the entire simulation, the choice is limited by the minimum breakpoint [35] distance h_{upper} among all the input sources. Fig. 1 (a) has 10ps as the upper limit for h in BE-FTS and TR-FTS. When the alignments of inputs change (as shown in Fig. 1 (b)) shift by 5ps, the resulting upper limit for h becomes 5ps for the approaches with fixed step size. If h is larger than the limit, it is impossible to guarantee the accuracy since we may skip pivot points of the inputs.

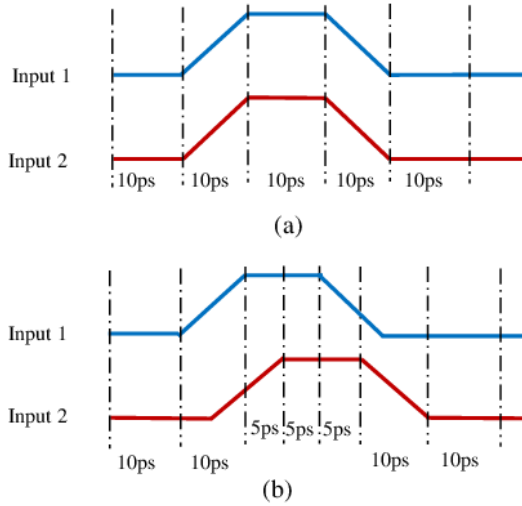


Fig. 1. Example: Interleave two input sources to create smaller transition time. (a) Before interleaving, the smallest transition time of the input sources is $h_{upper} = 10ps$; (b) After interleaving, the smallest transition time of the input sources is $h_{upper} = 5ps$.

C. Matrix Exponential Time Integration Scheme

The solution of Eq. (1) can be obtained analytically [29]. For a simple illustration, we convert Eq. (1) into

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}(t), \quad (4)$$

when \mathbf{C} is not singular¹,

$$\mathbf{A} = -\mathbf{C}^{-1}\mathbf{G}, \text{ and } \mathbf{b}(t) = \mathbf{C}^{-1}\mathbf{B}\mathbf{u}(t).$$

Given a solution at time t and a time step h , the solution at $t + h$ is

$$\mathbf{x}(t+h) = e^{h\mathbf{A}}\mathbf{x}(t) + \int_0^h e^{(h-\tau)\mathbf{A}}\mathbf{b}(t+\tau)d\tau. \quad (5)$$

Assuming that the input $\mathbf{u}(t)$ is a piecewise linear (PWL) function of t , we can integrate the last term of Eq. (5) analytically,

¹The assumption is to simplify the explanation in this section. After Sec. III-B, we use I-MATEX, R-MATEX and DR-MATEX to compute the solution of DAE without inversion of \mathbf{C} . Therefore, the methods are suitable for general DAE system, i.e., Eq. (1) without the assumption here.

ically, turning the solution with matrix exponential operator:

$$\mathbf{x}(t+h) = -\left(\mathbf{A}^{-1}\mathbf{b}(t+h) + \mathbf{A}^{-2}\frac{\mathbf{b}(t+h) - \mathbf{b}(t)}{h}\right) + e^{h\mathbf{A}}\left(\mathbf{x}(t) + \mathbf{A}^{-1}\mathbf{b}(t) + \mathbf{A}^{-2}\frac{\mathbf{b}(t+h) - \mathbf{b}(t)}{h}\right). \quad (6)$$

For the time step choice, breakpoints (also known as input transition spots (TS) [47]) are the time points where slopes of input vector change. Therefore, for Eq. (6), the maximum time step starting from t is $(t_s - t)$, where t_s is the smallest one in TS larger than t . In matrix exponential based framework, the limitation of time step size is not the local truncation error (LTE), but the activities among all input sources.

III. KRYLOV SUBSPACE METHODS FOR MATRIX EXPONENTIAL AND VECTOR PRODUCT (MEVP)

In PDN simulation, \mathbf{A} is usually above millions and makes the direct computation of matrix exponential $e^{\mathbf{A}}$ infeasible. The alternative way to compute the product is through Krylov subspace method [37]. In this section, we first introduce the background of standard Krylov subspace for MEVP. Then, we discuss invert (I-MATEX) and rational Krylov subspace (R-MATEX) methods, which highly improve the runtime performance for MEVP.

A. MEXP: MEVP Computation via Standard Krylov Subspace Method

The complexity of $e^{\mathbf{A}}\mathbf{v}$ can be reduced using Krylov subspace method and still maintained in a high order polynomial approximation [37]. MEXP [36] uses standard Krylov subspace, which uses \mathbf{A} directly to generate subspace basis through Arnoldi process (Algorithm 1). First, we reformulate Eq. (6) into

$$\mathbf{x}(t+h) = e^{h\mathbf{A}}(\mathbf{x}(t) + \mathbf{F}(t,h)) - \mathbf{P}(t,h), \quad (7)$$

where

$$\mathbf{F}(t,h) = \mathbf{A}^{-1}\mathbf{b}(t) + \mathbf{A}^{-2}\frac{\mathbf{b}(t+h) - \mathbf{b}(t)}{h} \quad (8)$$

and

$$\mathbf{P}(t,h) = \mathbf{A}^{-1}\mathbf{b}(t+h) + \mathbf{A}^{-2}\frac{\mathbf{b}(t+h) - \mathbf{b}(t)}{h}. \quad (9)$$

The standard Krylov subspace

$$\mathbf{K}_m(\mathbf{A}, \mathbf{v}) := \text{span}\{\mathbf{v}, \mathbf{A}\mathbf{v}, \dots, \mathbf{A}^{m-1}\mathbf{v}\} \quad (10)$$

obtained by Arnoldi process has the relation

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^T, \quad (11)$$

where \mathbf{H}_m is the upper Hessenberg matrix

$$\mathbf{H}_m = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,m-1} & h_{2,m} \\ 0 & h_{3,2} & \cdots & h_{3,m-1} & h_{3,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & h_{m,m-1} & h_{m,m} \end{pmatrix}, \quad (12)$$

\mathbf{V}_m is a $n \times m$ matrix by $(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m)$, and \mathbf{e}_m is the m -th unit vector with dimension $n \times 1$. MEVP is computed via

$$e^{h\mathbf{A}}\mathbf{v} \approx \beta \mathbf{V}_m e^{h\mathbf{H}_m} \mathbf{e}_1. \quad (13)$$

The posterior error term is

$$r_m(h) = \|\beta h_{m+1,m} \mathbf{v}_{m+1} \mathbf{e}^T e^{h\mathbf{H}_m} \mathbf{e}_1\|, \quad (14)$$

where $\beta = \|\mathbf{v}\|$. However, for an autonomous system $\mathbf{C}\dot{\mathbf{x}}(t) = -\mathbf{G}\mathbf{x}(t)$ in circuit simulation, we consider the residual between $\mathbf{C}\dot{\mathbf{x}}(t)$ and $-\mathbf{G}\mathbf{x}(t)$, which is

$$\mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{G}\mathbf{x}(t),$$

instead of

$$\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t)$$

in $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$. This leads to

$$r(m, h) = \|\beta h_{m+1,m} \mathbf{C}\mathbf{v}_{m+1} \mathbf{e}^T e^{h\mathbf{H}_m} \mathbf{e}_1\| \quad (15)$$

and helps us mitigate the overestimation of the error bound.

To generate $\mathbf{x}(t+h)$ by Algorithm 1, we use

$$[\mathbf{L}, \mathbf{U}] = \text{LU_Decompose}(\mathbf{X}_1), \quad (16)$$

where

$$\mathbf{X}_1 = \mathbf{C} \text{ and } \mathbf{X}_2 = \mathbf{G}$$

as inputs for standard Krylov subspace. The error budget ϵ and Eq. (14) are used to determine the convergence when time step is h and Krylov subspace dimension is j (from line 11 to line 14 in Algorithm 1).

Algorithm 1: MATEX_Arnoldi

Input: $\mathbf{L}, \mathbf{U}, \mathbf{X}_2, h, t, \mathbf{x}(t), \epsilon, \mathbf{P}(t, h), \mathbf{F}(t, h)$

Output: $\mathbf{x}(t+h), \mathbf{V}_m, \mathbf{H}, \mathbf{v}$

```

1  $\mathbf{v} = \mathbf{x}(t) + \mathbf{F}(t, h);$ 
2  $\mathbf{v}_1 = \frac{\mathbf{v}}{\|\mathbf{v}\|};$ 
3 for  $j = 1 : m$  do
4    $\mathbf{w} = \mathbf{U} \setminus (\mathbf{L} \setminus (\mathbf{X}_2 \mathbf{v}_j));$  /* a pair of forward
   and backward substitutions.          */
5   for  $i = 1 : j$  do
6      $h_{i,j} = \mathbf{w}^T \mathbf{v}_i;$ 
7      $\mathbf{w} = \mathbf{w} - h_{i,j} \mathbf{v}_i;$ 
8   end
9    $h_{j+1,j} = \|\mathbf{w}\|;$ 
10   $\mathbf{v}_{j+1} = \frac{\mathbf{w}}{h_{j+1,j}};$ 
11  if  $r(j, h) < \epsilon$  then
12     $m = j;$ 
13    break;
14  end
15 end
16  $\mathbf{x}(t+h) = \|\mathbf{v}\| \mathbf{V}_m e^{h\mathbf{H}} \mathbf{e}_1 - \mathbf{P}(t, h);$ 
```

The standard Krylov subspace may not be efficient when simulating stiff circuits [36], [49]. For the accuracy of approximation of $e^{\mathbf{A}}\mathbf{v}$, a large dimension of Krylov subspace basis is required, which not only brings the computational complexity but also consumes huge amount of memory. The

reason is that the Hessenberg matrix \mathbf{H}_m of standard Krylov subspace tends to approximate the large magnitude eigenvalues of \mathbf{A} [50]. Due to the exponential decay of higher order terms in Taylor's expansion, such components are not the crux of circuit system's behavior [50], [51]. Therefore, to simulate stiff circuit, we need to gather more vectors into subspace basis and increase the size of \mathbf{H}_m to fetch more useful components, which results in both memory overhead and computational complexity to Krylov subspace generations for each time step. In the following subsections, we adopt ideas from *spectral transformation* [50], [51] to effectively capture small magnitude eigenvalues in \mathbf{A} , leading to a fast and accurate MEVP computation.

B. I-MATEX: MEVP Computation via Invert Krylov Subspace Method

Instead of \mathbf{A} , we use \mathbf{A}^{-1} (or $\mathbf{G}^{-1}\mathbf{C}$) as our target matrix to form

$$\mathbf{K}_m(\mathbf{A}^{-1}, \mathbf{v}) := \text{span}\{\mathbf{v}, \mathbf{A}^{-1}\mathbf{v}, \dots, \mathbf{A}^{-(m-1)}\mathbf{v}\}. \quad (17)$$

Intuitively, by inverting \mathbf{A} , the small magnitude eigenvalues become the large ones of \mathbf{A}^{-1} . The resulting \mathbf{H}_m is likely to capture these eigenvalues first. Based on Arnoldi algorithm, the invert Krylov subspace has the relation

$$\mathbf{A}^{-1}\mathbf{V}_m = \mathbf{V}_m \mathbf{H}_m + h_{m+1,m} \mathbf{v}_{m+1} \mathbf{e}_m^T. \quad (18)$$

The matrix exponential $e^{\mathbf{A}}\mathbf{v}$ is calculated as

$$e^{\mathbf{A}}\mathbf{v} \approx \beta \mathbf{V}_m e^{h\mathbf{H}_m^{-1}} \mathbf{e}_1. \quad (19)$$

To put this method into Algorithm 1 is just by modifying the inputs $\mathbf{X}_1 = \mathbf{G}$ for the LU decomposition in Eq. (16), and $\mathbf{X}_2 = \mathbf{C}$. In the line 16 of Algorithm 1,

$$\mathbf{H} = \mathbf{H}_m^{-1}$$

for the invert Krylov version. The posterior error approximation [47] is

$$r_m(h) = \|\beta h_{m+1,m} \mathbf{A}\mathbf{v}_{m+1} \mathbf{e}_m^T \mathbf{H}_m^{-1} e^{h\mathbf{H}} \mathbf{e}_1\|, \quad (20)$$

which is derived from residual based error approximation in [51]. However, as mentioned in Sec. III-A, we consider the residual of $(\mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{G}\mathbf{x}(t))$, instead of $(\dot{\mathbf{x}}(t) - \mathbf{A}\mathbf{x}(t))$, which leads to

$$r(m, h) = \|\beta h_{m+1,m} \mathbf{G}\mathbf{v}_{m+1} \mathbf{e}_m^T \mathbf{H}_m^{-1} e^{h\mathbf{H}_m} \mathbf{e}_1\|. \quad (21)$$

We use Eq. (21) for the line 11 of Alg. 1.

C. R-MATEX: MEVP Computation via Rational Krylov Subspace Method

The shift-and-invert Krylov subspace basis [50] is designed to confine the spectrum of \mathbf{A} . Then, we generate Krylov subspace via

$$\mathbf{K}_m((\mathbf{I} - \gamma\mathbf{A})^{-1}, \mathbf{v}) := \text{span}\{\mathbf{v}, (\mathbf{I} - \gamma\mathbf{A})^{-1}\mathbf{v}, \dots, (\mathbf{I} - \gamma\mathbf{A})^{-(m-1)}\mathbf{v}\}, \quad (22)$$

where γ is a predefined parameter. With this shift, all the eigenvalues' magnitudes are larger than one. Then the inverse

limits the magnitudes smaller than one. According to [50], [51], the shift-and-invert basis for matrix exponential based transient simulation is not very sensitive to γ , once it is set to around the order near time steps used in transient simulation. The similar idea has been applied to simple power grid simulation with matrix exponential method [52]. Here, we generalize this technique and integrate into MATEX. The Arnoldi process constructs \mathbf{V}_m and \mathbf{H}_m . We have

$$(\mathbf{I} - \gamma\mathbf{A})^{-1}\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^T. \quad (23)$$

We can project the $e^{\mathbf{A}h}$ onto the rational Krylov subspace as follows.

$$e^{\mathbf{A}h}\mathbf{v} \approx \beta\mathbf{V}_m e^{h\frac{\mathbf{I}-\mathbf{H}_m^{-1}}{\gamma}}\mathbf{e}_1. \quad (24)$$

In the line 16 of Algorithm 1,

$$\mathbf{H} = \frac{\mathbf{I} - \mathbf{H}_m^{-1}}{\gamma}.$$

Following the same procedure [47], [51], the posterior error approximation is derived as

$$r_m(h) = \|\beta h_{m+1,m} \frac{\mathbf{I} - \gamma\mathbf{A}_m}{\gamma} \mathbf{v}_{m+1} \mathbf{e}_m^T \mathbf{H}_m^{-1} e^{h\mathbf{H}} \mathbf{e}_1\|. \quad (25)$$

Note that in practice, instead of computing $(\mathbf{I} - \gamma\mathbf{A})^{-1}$ directly, $(\mathbf{C} + \gamma\mathbf{G})^{-1}\mathbf{C}$ is utilized. The corresponding Arnoldi process shares the same skeleton of Algorithm 1 with input matrices

$$\mathbf{X}_1 = (\mathbf{C} + \gamma\mathbf{G})$$

for the LU decomposition Eq. (16), and

$$\mathbf{X}_2 = \mathbf{C}.$$

The residual estimation is

$$r(m, h) = \|\beta h_{m+1,m} \frac{\mathbf{C} + \gamma\mathbf{G}}{\gamma} \mathbf{v}_{m+1} \mathbf{e}_m^T \mathbf{H}_m^{-1} e^{h\mathbf{H}_m} \mathbf{e}_1\|. \quad (26)$$

Then, we plug Eq. (26) into the line 11 of Algorithm 1.

D. Regularization-Free MEVP Computation

When \mathbf{C} is a singular matrix, MEXP [36] needs the regularization process [53] to remove the singularity of DAE in Eq. (1). It is because MEXP needs factorize \mathbf{C} directly to form the input \mathbf{X}_1 for Algorithm 1. This brings extra computational overhead when the case is large [53]. It is not necessary if we can obtain the generalized eigenvalues and corresponding eigenvectors for matrix pencil $(-\mathbf{G}, \mathbf{C})$. Based on [54], we derive the following lemma,

Lemma 1. *Considering a homogeneous system*

$$\mathbf{C}\dot{\mathbf{x}} = -\mathbf{G}\mathbf{x}.$$

\mathbf{u} and λ are the eigenvector and eigenvalue of matrix pencil $(-\mathbf{G}, \mathbf{C})$, then

$$\mathbf{x} = e^{t\lambda}\mathbf{u}$$

is a solution of the system.

Proof. ² If λ and \mathbf{u} are an eigenvalue and eigenvector of a generalized eigenvalue problem

$$-\mathbf{G}\mathbf{u} = \lambda\mathbf{C}\mathbf{u}.$$

Then, $\mathbf{x} = e^{t\lambda}\mathbf{u}$ is the solution of $\mathbf{C}\dot{\mathbf{x}} = -\mathbf{G}\mathbf{x}$. \square

Because we do not need to compute \mathbf{C}^{-1} explicitly during Krylov subspace generation, I-MATEX and R-MATEX are regularization-free. Instead, we factorize \mathbf{G} for invert Krylov subspace basis generation (I-MATEX), or $(\mathbf{C} + \gamma\mathbf{G})$ for rational Krylov subspace basis (R-MATEX).³ Besides, their Hessenberg matrices Eq. (12) are invertible, which contain corresponding important generalized eigenvalues/eigenvectors from matrix pencil $(-\mathbf{G}, \mathbf{C})$, and define the behavior of linear dynamic system in Eq. (1) of interest.

E. Comparisons among Different Krylov Subspace Algorithms for MEVP Computation

In order to observe the error distribution versus dimensions of standard, invert, and rational Krylov subspace methods for MEVP, we construct a RC circuit with stiffness

$$\frac{Re(\lambda_{min})}{Re(\lambda_{max})} = 4.7 \times 10^6,$$

where $\lambda_{max} = -8.49 \times 10^{10}$ and $\lambda_{min} = -3.98 \times 10^{17}$ are the maximum and minimum eigenvalues of $\mathbf{A} = -\mathbf{C}^{-1}\mathbf{G}$. Fig. 2 shows the relative error reductions along the increasing Krylov subspace dimension. The error reduction rate of rational Krylov subspace is the best, while the one of standard Krylov subspace requires huge dimension to capture the same level of error. For example, it costs almost 10 \times of the size to achieve around relative error 1% compared to Invert and Rational Krylov subspace methods. The relative error is

$$\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{h\mathbf{H}_m}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|},$$

where $h = 0.4ps$, $\gamma = 10^{-13}$. The matrix \mathbf{A} is a relatively small matrix and computed by MATLAB *expm* function. The result of $e^{h\mathbf{A}}\mathbf{v}$ serves as the baseline for accuracy. The relative error is the real relative difference compared to the analytical solution $e^{h\mathbf{A}}\mathbf{v}$ of the ODE

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}$$

with an initial vector \mathbf{v} , which is generated by MATLAB *rand* function.

The error reduction rate of standard Krylov subspace is the worst, while the rational Krylov subspace is the best. It is the reason that we prefer rational Krylov subspace (R-MATEX). The relative errors of BE, TR and FE are 0.0594, 0.4628, and 2.0701×10^4 , respectively. The large error of FE is due to the instability issue of its low order explicit time integration scheme. In Fig. 2, when $m = 3$, standard, invert and rational Krylov subspace methods have 0.8465, 0.0175, and 0.0065, respectively. It illustrates the power of matrix exponential

²We repeat the proof from [54] with some modifications for our formulation.

³It is also applied to the later work of DR-MATEX in Sec. IV-B.

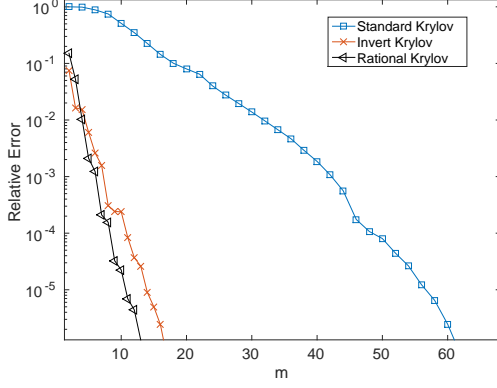


Fig. 2. The relative error vs. dimensional m of different Krylov subspace methods. The relative error is $\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{h\mathbf{H}_m}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$, where $h = 0.4ps$, $\gamma = 10^{-13}$. Note: The relative error is the difference compared to analytical solution $e^{h\mathbf{A}}\mathbf{v}$ of the ODE $\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}$ with an initial vector \mathbf{v} , which is generated by MATLAB *rand* function, and its entries are positive numbers in $(0, 1]$.

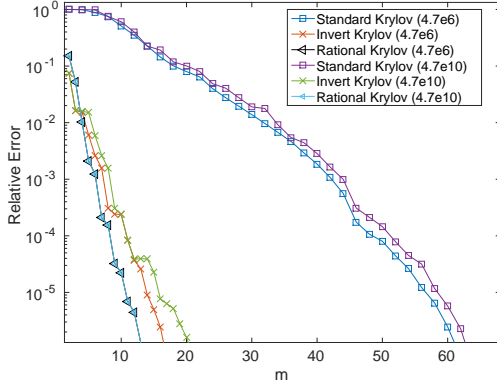


Fig. 3. The relative error vs. dimension m of different Krylov subspace methods. The relative error is $\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{h\mathbf{H}_m}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$, where $h = 0.4ps$, $\gamma = 10^{-13}$. The rational Krylov subspace has very stable error reduction rate. The number in the bracket represents the stiffness value of the system.

method. Our proposed methods are all stable and can achieve improved error numbers.

In order to observe the different stiffness effects on Krylov subspace methods, we change the entries in \mathbf{C} and \mathbf{G} to make the different stiffness value 4.7×10^{10} . Fig. 3 illustrates the stable reduction rate of rational method. The stiffness degrades the performance of standard Krylov subspace method. Both invert and rational Krylov subspace methods are good candidates for stiff circuit system.

Regarding the relative error distributions vs. time step h and dimension m , Fig. 4, Fig. 5, and Fig. 6 are computed by standard, invert, and rational Krylov subspaces ($\gamma = 5 \times 10^{-13}$), respectively. Fig. 4 shows that the errors generated by standard Krylov subspace method has flat region with high error values in time-step range of interests. The small ('unrealistic') time step range has small error values. Compared to Fig. 4, invert (Fig. 5) and rational (Fig. 6) Krylov subspace methods reduce errors quickly for large h . The explanation is that a relatively

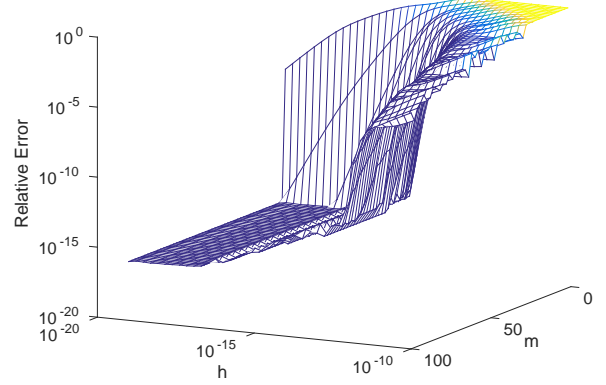


Fig. 4. The error of MEVP via standard Krylov Subspace: $\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{h\mathbf{H}_m}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$ vs. time step h and dimension of standard Krylov subspace basis (m). The standard Krylov subspace approximates the solution well in extremely small h , since it captures the important eigenvalues and eigenvectors of \mathbf{A} at that region. However, the small h is not useful for the circuit simulation. For large h , it costs large m to reduce the error.

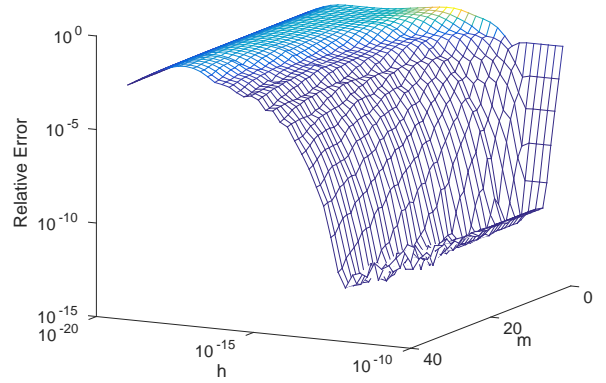


Fig. 5. The error of MEVP via invert Krylov Subspace: $\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{h\mathbf{H}_m^{-1}}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$ vs. time step h and dimension of invert Krylov subspace basis (m). Compared to Fig. 4, invert Krylov subspace method reduces the errors for large h .

small portion of the eigenvalues and corresponding invariant subspaces determines the final result (vector) when time step h is larger [50], which are efficiently captured by invert and rational Krylov subspace methods.

The error of rational Krylov subspace is relatively insensitive to γ when it is selected between the time-step range of interests (Fig. 7). Above all, rational Krylov (R-MATEX) and invert Krylov (I-MATEX) subspace methods have much better performance than standard version. When we deal with stiff cases, standard Krylov subspace is not a feasible choice due to the large dimension m of Krylov subspace, which causes huge memory consumption and poor runtime performance.

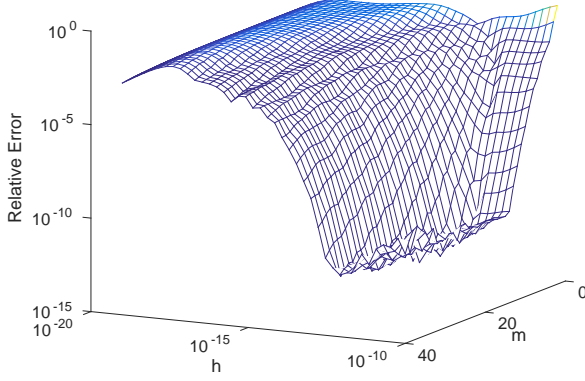


Fig. 6. The error of MEVP via rational Krylov Subspace:

$\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{\frac{\mathbf{I} - \mathbf{H}_m^{-1}}{\gamma}}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$, where $\gamma = 5 \times 10^{-13}$, vs. time step h and dimension of rational Krylov subspace basis (m). Compared to Fig. 4, rational Krylov subspace method reduces the errors for large h as Fig. 5.

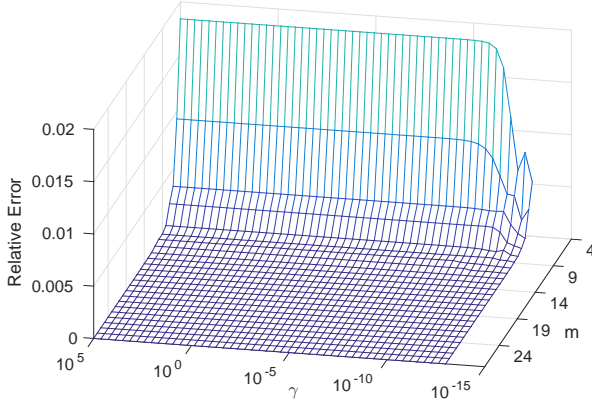


Fig. 7. The error of MEVP via rational Krylov Subspace $\frac{\|e^{h\mathbf{A}}\mathbf{v} - \beta\mathbf{V}_m e^{\frac{\mathbf{I} - \mathbf{H}_m^{-1}}{\gamma}}\mathbf{e}_1\|}{\|e^{h\mathbf{A}}\mathbf{v}\|}$, where $h = 4ps$. The flat region shows the error is actually relatively insensitive to γ , when γ is in the range of step size h of interests.

IV. MATEX FRAMEWORK

A. MATEX Circuit Solver

We incorporate matrix exponential based integration scheme with Krylov subspace method into our MATEX framework, which is summarized in Algorithm 2. We set \mathbf{X}_1 and \mathbf{X}_2 in Line 1 based on the choice of Krylov subspace method as follows,

- I-MATEX: $\mathbf{X}_1 = \mathbf{G}$, $\mathbf{X}_2 = \mathbf{C}$
- R-MATEX: $\mathbf{X}_1 = \mathbf{C} + \gamma\mathbf{G}$, $\mathbf{X}_2 = \mathbf{C}$

For linear system of PDN, the matrix factorization in line 4 is only performed once, and the matrices \mathbf{L} and \mathbf{U} are reused in the while loop from line 5 to line 10. Line 8 uses Arnoldi process with corresponding inputs to construct Krylov subspace as shown in Algorithm 1.

Algorithm 2: MATEX Circuit Solver

Input: $\mathbf{C}, \mathbf{G}, \mathbf{B}, \mathbf{u}, \epsilon$, and time span T .
Output: The set of \mathbf{x} from $[0, T]$.

- 1 Set $\mathbf{X}_1, \mathbf{X}_2$;
- 2 $t = 0$;
- 3 $\mathbf{x}(t) = \text{DC_analysis}$;
- 4 $[\mathbf{L}, \mathbf{U}] = \text{LU_Decompose}(\mathbf{X}_1)$;
- 5 **while** $t < T$ **do**
- 6 Compute maximum allowed step size h ;
- 7 Update $\mathbf{P}(t, h), \mathbf{F}(t, h)$;
- 8 Obtain $\mathbf{x}(t + h)$ by **Algorithm 1** with inputs $[\mathbf{L}, \mathbf{U}, \mathbf{X}_2, h, t, \mathbf{x}(t), \epsilon, \mathbf{P}(t, h), \mathbf{F}(t, h)]$;
- 9 $t = t + h$;
- 10 **end**

B. DR-MATEX (Distributed R-MATEX Framework) by Decomposition of Input Sources, Linear Superposition, and Parallel Computation Model

1) Motivation:

There are usually many input sources in PDNs as well as their transition activities, which might narrow the regions for the stepping of matrix exponential based method due to the unaligned breakpoints. In other words, the region before the next transition t_s may be shortened when there are a lot of activities from the input sources. It leads to more chances of generating new Krylov subspace bases. We want to reduce the number of subspace generations and improve the runtime performance.⁴

2) Treatment and Methodology:

In matrix exponential based integration framework, we can choose any time spot $t + h \in [t, t_s]$ with computed Krylov subspace basis. The solution of $\mathbf{x}(t + h)$ is computed by scaling the existing Hessenberg matrix \mathbf{H} with the time step h as below

$$\mathbf{x}(t + h) = \|\mathbf{v}\| \mathbf{V}_m e^{h\mathbf{H}} \mathbf{e}_1 - \mathbf{P}(t, h). \quad (27)$$

This is an important feature for computing the solutions at intermediate time points without generating the Krylov subspace basis, when there is no current transition. Besides, since the PDN is linear dynamical system, we can utilize the well-known superposition property of linear system and distributed computing model to tackle this challenge.

To illustrate our distributed version of MATEX framework, we first define three terms to categorize the breakpoints of input sources:

- *Local Transition Spot (LTS)*: the set of TS at an input source to the PDN.
- *Global Transition Spot (GTS)*: the union of LTS among all the input sources to the PDN.
- *Snapshot*: a set $GTS \setminus LTS$ at one input source.

If we simulate the PDN with respect to all the input sources, the points in the set of GTS are the places where generations

⁴The breakpoints also put the same constraint on TR-FTS and BE-FTS. However, their time steps are fixed already, which refrains them from reaching this problem in the first place.

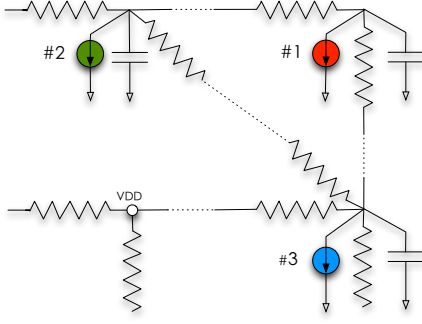


Fig. 8. Part of a PDN model with input sources from Fig. 9.

of Krylov subspace cannot be avoided. For example, there are three input sources in a PDN (Fig. 8). The input waveforms are shown in Fig. 9. The first line is *GTS*, which is contributed by the union of *LTS* in input sources #1, #2 and #3. However, we can partition the task into subtasks by simulating each input source individually. Then, each subtask generates Krylov subspaces based on its own *LTS* and keeps track of *Snapshot* for the later usage of summation via linear superposition. Between two *LTS* points t and $t + h$, the *Snapshot* points

$$t + h_1 < t + h_2 < \dots < t + h_l \in (t, t + h]$$

can reuse the Krylov subspace generated at t . For each node, the chances of generation of Krylov subspaces are reduced. The time periods of reusing latest Krylov subspaces are enlarged locally and bring the runtime improvement. Besides, when subtasks are assigned, there is no communication among the computing nodes, which leads to so-called *Embarrassingly Parallel* computation model.

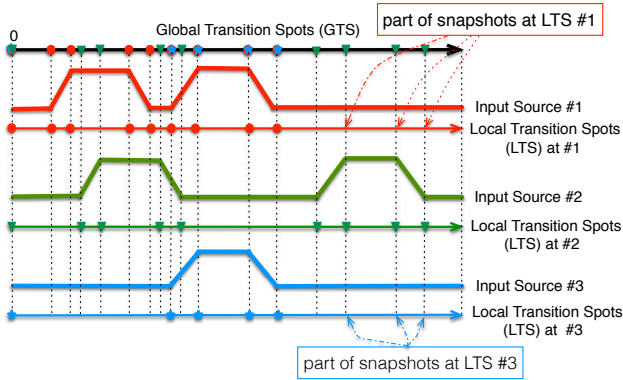


Fig. 9. Illustration of input transitions. *GTS*: Global Transition Spots; *LTS*: Local Transition Spots; *Snapshots*: the crossing positions by dash lines and *LTS* # k without solid points.

3) *More Aggressive Tasks Decomposition*: We divide the simulation task based on the alignments of input sources. More aggressively, we can decompose the task according to the “bump” shapes of the input sources.⁵ We group the input sources, which have the same

$$(t_{\text{delay}}, t_{\text{rise}}, t_{\text{fall}}, t_{\text{width}})$$

⁵IBM power grid benchmarks provide the pulse input model in SPICE format.

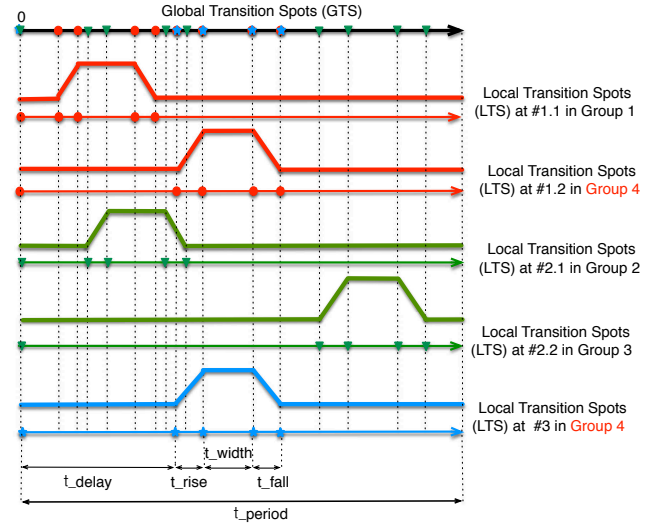


Fig. 10. Grouping of “Bump” shape transitions for sub-task simulation. The matrix exponential based method can utilize adaptive stepping in each *LTS* and reuse the Krylov subspace basis generated at the latest point in *LTS*. However, traditional methods (TR, BE, etc.) still need to do time marching, either by pairs of forward and backward substitutions and proceed with fixed time step, or by re-factorizing matrix and solving linear system for adaptive stepping. (Pulse input information: t_{delay} : delay time; t_{rise} : rise time; t_{width} : pulse width; t_{fall} : fall time; and t_{period} : period).

into one set. For example, the input source #1 of Fig. 9 is divided to #1.1 and #1.2 in Fig. 10. The input source #2 in Fig. 9 is divided to #2.1 and #2.2 in Fig. 10. Therefore, there are four groups in Fig. 10, Group 1 contains *LTS*#1.1. Group 2 contains *LTS*#2.1. Group 3 contains *LTS*#2.2. Group 4 contains *LTS*#1.2 and #3. Our proposed framework MATEX is shown in Fig. 11. After pre-computing *GTS* and decomposing *LTS* based on “bump” shape (Fig. 10), we group them and form *LTS* #1 ~ # K .⁶

4) MATEX Scheduler in DR-MATEX:

In DR-MATEX, the role of MATEX scheduler is just to send out *GTS* and *LTS* to different MATEX slave nodes and collect final results after all the subtasks of transient simulation are finished. The node number is based on the total number of subtasks, which is the group number after PDN source decomposition. Then the simulation computations are performed in parallel. Each node has its own inputs. For example, Node# k has *GTS*, *LTS*# k , \mathbf{P}_k and \mathbf{F}_k , which contain the corresponding \mathbf{b} for node k . Scheduler does not need to do anything during the transient simulation, since there are no communications among nodes before the stage of “write back” (in Fig. 11), by when all nodes complete their transient simulations.

Within each slave node, the circuit solver (Algorithm 3) computes transient response with varied time steps. Solutions are obtained without re-factorizing matrix during the computation of transient simulation. The computing nodes write back the results and inform the MATEX scheduler after finishing their own transient simulation.

⁶There are alternative decomposition strategies. It is also easy to extend the work to deal with different input waveforms. We try to keep this part as simple as possible to emphasize our framework.

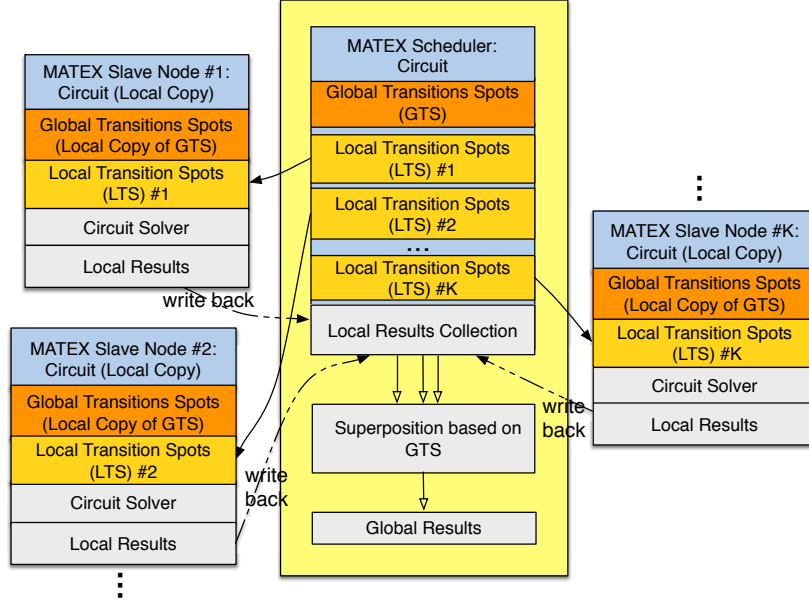


Fig. 11. DR-MATEX: The distributed MATEX framework using R-MATEX circuit solver.

Algorithm 3: DR-MATEX: The distributed MATEX framework using R-MATEX at Node# k .

Input: $LTS\#k$, GTS , P_k , F_k , error tolerance E_{tol} , and simulation time span T .

Output: Local solution \mathbf{x} along GTS in node

$k \in [1, \dots, S]$, where S is the number of nodes

```

1  $t = 0$ ,  $\mathbf{X}_1 = \mathbf{C} + \gamma\mathbf{G}$ , and  $\mathbf{X}_2 = \mathbf{C}$ ;
2  $\mathbf{x}(t) = \text{Local\_Initial\_Solution}$ ;
3  $[\mathbf{L}, \mathbf{U}] = \text{LU\_Decompose}(\mathbf{X}_1)$ ;
4 while  $t < T$  do
5   Compute maximum allowed step size  $h$  based on
    $GTS$ ;
6   if  $t \in LTS\#k$  then
7     /* Generate Krylov subspace for the
       point at  $LTS\#k$  and compute  $\mathbf{x}(t+h)$  */
8      $[\mathbf{x}(t+h), \mathbf{V}_m, \mathbf{H}_m, \mathbf{v}] =$ 
      MATEX_Arnoldi( $\mathbf{L}, \mathbf{U}, \mathbf{X}_2,$ 
       $h, t, \mathbf{x}(t), \epsilon, \mathbf{P}_k(t, h), \mathbf{F}_k(t, h)$ );
9      $a_{lts} = t$ ;
10    end
11  else
12    /* Obtain  $\mathbf{x}(t+h)$  at Snapshot with
       computed Krylov subspace */
13     $h_a = t + h - a_{lts}$ ;
14     $\mathbf{x}(t+h) = \|\mathbf{v}\| \mathbf{V}_m e^{h_a \mathbf{H}_m} \mathbf{e}_1 - \mathbf{P}_k(t, h)$ ;
15  end
16   $t = t + h$ ;
17 end

```

C. Runtime Analysis of MATEX PDN Solver

Suppose we have the dimension of Krylov subspace basis m on average for each time step and one pair of forward and backward substitutions consumes runtime T_{bs} . The total time of serial parts is T_{serial} , which includes matrix factorizations, result collection, etc. For $\mathbf{x}(t+h)$, the evaluation of matrix exponential with $e^{h\mathbf{H}_m}$ is T_H , which is in proportion to the time complexity $O(m^3)$. Besides, we need extra T_e to form $\mathbf{x}(t+h)$, which is proportional to $O(nm^2)$ by $\beta\mathbf{V}_m e^{h\mathbf{H}_m} \mathbf{e}_1$.

Given K points of GTS , without decomposition of input sources, the runtime is

$$KmT_{bs} + K(T_H + T_e) + T_{serial}. \quad (28)$$

After dividing the input transitions and sending to enough computing nodes, we have k points of LTS for each node based on feature extraction and grouping (e.g., $k = 4$ for one “bump” shape feature). The total computation runtime is

$$kmT_{bs} + K(T_H + T_e) + T_{serial}, \quad (29)$$

where $K(T_H + T_e)$ contains the portion of computing *Snapshot* in DR-MATEX mode. The speedup of DR-MATEX over single MATEX is

$$\text{Speedup} = \frac{KmT_{bs} + K(T_H + T_e) + T_{serial}}{kmT_{bs} + K(T_H + T_e) + T_{serial}}. \quad (30)$$

For R-MATEX, we have small m . Besides, T_{bs} is relatively larger than $(T_H + T_e)$ in our targeted problem. Therefore, the most dominating part is the KmT_{bs} in Eq. (28). We can always decompose input source transitions, and make k smaller than K .

In contrast, suppose the traditional method with fixed step size has N steps for the entire simulation, the runtime is

$$NT_{bs} + T_{serial}.$$

TABLE I
SPECIFICATIONS OF IBM POWER GRID BENCHMARKS.

Design	#R	#C	#L	#I	#V	#Nodes
ibmpg1t	41K	11K	277	11K	14K	54K
ibmpg2t	245K	37K	330	37K	330	165K
ibmpg3t	1.6M	201K	955	201K	955	1.0M
ibmpg4t	1.8M	266K	962	266K	962	1.2M
ibmpg5t	1.6M	473K	277	473K	539K	2.1M
ibmpg6t	2.4M	761K	381	761K	836K	3.2M

Then, the speedup of distributed DR-MATEX over the traditional method is

$$\text{Speedup}' = \frac{NT_{bs} + T_{serial}}{kmT_{bs} + K(T_H + T_e) + T_{serial}}. \quad (31)$$

Note that, when the minimum distance among input source breakpoints decreases, large time span or many cycles is required to simulate PDNs, the schemes with such uniform step size would degrade runtime performance furthermore due to the increase of N . In contrast, in MATEX PDN solver, K is not so sensitive to such constraints. Besides, k can be maintained in a small number based on the decomposition strategy. Therefore, the speedups of our proposed methods tend to be larger when the simulation requirements become harsher.

V. EXPERIMENTAL RESULTS

We implement all the algorithms in MATLAB R2014b⁷ and use UMFPACK package for LU factorization. First, we compare I-MATEX, R-MATEX and TR in order to show our runtime improvements in single machine framework in Table II. Second, we show our distributed framework DR-MATEX achieves large speedups in Table III. The experiments are conducted on the server with Intel(R) Xeon (R) E5-2640 v3 2.60GHz processor and 125GB memory.

A. Performance of I-MATEX and R-MATEX in Sec. IV-A

We compare our proposed I-MATEX and R-MATEX against the popular TR-FTS on the IBM power grid benchmarks [22]. Among the current sources, the smallest interval between two breakpoints is $h_{upper} = 10ps$, which puts the upper limit of the TR's step size. All of these cases have very large numbers of input current sources. Table I shows the details of each benchmark circuit of which size ranges from 54K up to 3.2M. The simulation time is $10ns$. From ibmpg1t to ibmpg6t, TR uses fixed step size in $10ps$. We also change the IBM power grid benchmark to make the smallest distance among breakpoints $1ps$ by interleaving input sources' breakpoints (similar as Fig. 1). Therefore, the fixed step size method can only use at most $1ps$. The names of those benchmarks are ibmpg1t_new, ibmpg2t_new, ibmpg3t_new, ibmpg4t_new, ibmpg5t_new and ibmpg6t_new.

After DC analysis in TR-FTS, we LU factorize matrix once for the later transient simulation, which only contains

time stepping. Actually, multiple factorized matrices can be deployed [55], [56]. We can choose one of them during the stepping. The problem is the memory and runtime overhead for the multiple matrix factorizations. Another point is if large time step h' is chosen, the standard low order scheme cannot maintain the accuracy.

Experiment is conducted on a single computing node. In Table II, we record the total simulation runtime **Total(s)**, which includes the processes of DC and transient simulation, but excludes the non-numerical computation before DC, e.g., netlist parsing and matrix stamping. We also record the part of transient simulation **Tran(s)**, excluding DC analysis and LU decompositions. The speedup of I-MATEX is not as large as R-MATEX, because I-MATEX with a large spectrum of \mathbf{A} generates large dimension m of Krylov subspace. Meanwhile, the step size is not large enough to let it fully harvest the gain from time marching with stepping. In contrast, R-MATEX needs small dimension numbers m of rational Krylov subspace, which ranges from 2 to 8 in those cases. Therefore, they can benefit from large time stepping, shown as $SPDP_{tr}^r$. For ibmpg4t, R-MATEX achieves maximum speedup resulted from the relatively small number of breakpoints in that benchmark, which is around 44 points, while the majority of others have over 140 points.

In Table II, our single mode R-MATEX achieves the average speedup $5\times$ over TR-FTS. Note the average speedup number of single mode R-MATEX over TR-FTS for the original IBM benchmark (ibmpg1t~ibmpg6t) is less than the speedup of the new test cases (ibmpg1t_new~ibmpg6t_new). As we mentioned before, ibmpg1t_new~ibmpg6t_new have harsher input constraints, making the available step size only $1ps$. Therefore, the adaptive stepping by R-MATEX is more beneficial to the runtime performance in ibmpg1t_new~ibmpg6t_new than ibmpg1t~ibmpg6t.

B. Performance of DR-MATEX in Sec. IV-B

We test our distributed DR-MATEX in the following experiments with the same IBM power grid benchmarks. These cases have many input transitions (GTS) that limit step sizes of R-MATEX. We divide the region before the computation of simulation. We decompose the input sources by the approach discussed in Sec. IV-B3 and obtain much fewer transitions of LTS for computing nodes. The original input source numbers are over ten thousand in the benchmarks. However, based on "bump" feature (as shown in Fig. 10), we obtain a fairly small numbers for each computing node, which is shown as *Group #* in Table III. (Now, the fact that hundred machines to process in parallel is quite normal [38], [57] in the industry.) We pre-compute GTS and LTS groups and assign sub-tasks to corresponding nodes⁸. MATEX scheduler is only responsible for simple superposition calculation at the end of simulation. Since the slave nodes are in charge of all the computing procedures (Fig. 11) for the computation of

⁷Measurements reported are on MATLAB implementations. They are subject to limitations and are not directly comparable to C++ implementations reported in literature such as [44].

⁸ Based on the feature of input sources available, the preprocessing is very efficient, which takes linear time complexity to obtain GTS , LTS and separates the sources into different groups.

TABLE II

PERFORMANCE COMPARISONS (SINGLE COMPUTING NODE): TR-FTS, I-MATEX, AND R-MATEX. **DC(s)**: RUNTIME OF DC ANALYSIS (SECONDS); m_I : THE MAXIMUM m OF KRYLOV SUBSPACE IN I-MATEX. **Tran(s)**: RUNTIME OF TRANSIENT SIMULATION AFTER DC (SECONDS), EXCLUDING THE MATRIX FACTORIZATION RUNTIME; **Total(s)**: RUNTIME OF OVERALL TRANSIENT SIMULATION (SECONDS); **Df(uV)**: MAXIMUM AND AVERAGE VOLTAGE DIFFERENCES COMPARED TO PROVIDED SOLUTIONS (uV); m_R : THE MAXIMUM m OF KRYLOV SUBSPACE IN R-MATEX **SPDP_{tr}^r**: SPEEDUP OF R-MATEX OVER TR-FTS WITH RESPECT TO **Tran(s)**; **SPDP_i^r**: SPEEDUP OF R-MATEX OVER I-MATEX WITH RESPECT TO **Tran(s)**.

Design	DC(s)	TR-FTS		I-MATEX				R-MATEX				Speedups	
		Tran(s)	Total(s)	m_I	Tran(s)	Total(s)	Df(uV)	m_R	Tran(s)	Total(s)	Df(uV)	SPDP _{tr} ^r	SPDP _i ^r
ibmpg1t	0.2	5.7	6.00	30	28.8	28.9	58\9.8	5	10.1	10.3	45\6.8	0.6×	2.9×
ibmpg2t	0.8	40.0	41.9	28	130.0	130.9	92\10.5	5	35.6	37.4	45\6.8	1.1×	3.7×
ibmpg3t	16.4	263.2	295.0	29	1102.5	1115.1	95\20.4	5	275.5	301.0	95\18.5	1.0×	4.0×
ibmpg4t	13.5	460.8	501.9	29	433.8	458.2	101\39.3	5	200.5	239.1	99\34.2	2.3×	2.2×
ibmpg5t	9.0	476.6	498.0	30	1934.4	1944.5	29\5.6	5	383.1	401.9	29\4.4	1.2×	5.0×
ibmpg6t	15.3	716.0	749.1	25	2698.9	2713.7	39\8.6	5	773.5	800.5	33\5.6	0.9×	3.5×
ibmpg1t_new	0.2	51.3	51.7	30	27.2	27.4	58\9.8	5	11.7	12.1	53\6.9	4.4×	2.3×
ibmpg2t_new	0.9	431.4	433.5	28	114.9	115.7	49\10.5	5	43.3	44.9	33\5.6	10.0×	2.7×
ibmpg3t_new	16.3	3716.5	3749.0	29	1219.3	1232.6	95\20.4	5	481.7	508.2	95\18.9	7.7×	2.5×
ibmpg4t_new	18.3	5044.6	5085.3	29	753.5	776.4	101\39.3	6	350.9	387.2	99\34.2	14.4×	2.1×
ibmpg5t_new	10.5	5065.9	5110.1	30	2494.0	2504.7	30\5.6	5	746.2	766.4	30\4.4	6.8×	3.3×
ibmpg6t_new	13.1	7015.3	7059.7	25	3647.9	3663.1	39\8.6	6	895.1	923.1	33\7.3	7.8×	4.1×
Average	—	—	—	—	—	—	65\15.7	—	—	—	57\12.8	5×	3×

their own transient simulation tasks, and have no communications with others, our framework falls into the category of *Embarrassingly Parallelism* model. We can easily emulate the multiple-node environment. We simulate each group using the command “matlab -singleCompThread” in our server. We record the runtime numbers for each process (slave nodes) and report the maximum runtime as the total runtime “Total(s)” of DR-MATEX in Table III. We also record “pure transient simulation” as “Tran(s)”, which is the maximum runtime of the counterparts among all computing nodes.

For TR-FTS, we use $h = 10ps$, so there are 1,000 pairs of forward and backward substitutions during the process of pure transient simulation for ibmpg1t~ibmpg6t; We use $h = 1ps$ for ibmpg1t_new~ibmpg6t_new. Therefore, we have 10,000 pairs of forward and backward substitutions for stepping. In DR-MATEX, the circuit solver uses R-MATEX with $\gamma = 10^{-10}$, which is set to sit among the order of varied time steps during the simulation (since Sec. III-E discusses the insensitivity of γ around the step size of interests). TR-FTS is not distributed because it has no gain by dividing the current source as we do for the DR-MATEX. TR-FTS cannot avoid the repeated pairs of forward and backward substitutions. Besides, adaptive stepping for TR-FTS only degrades the performance, since the process requires extra matrix factorizations.

In Table III, our distributed mode gains up to 98×

 for the pure transient computing. The average peak dimension m of rational Krylov subspace is 7. The memory overhead ratio for each node (around 1.6× over TR-FTS on average) is slightly larger, which is worthwhile with respect to the large runtime improvement. With the huge reduction of runtime for Krylov subspace generations, the serial parts, including LU and DC, play more dominant roles in DR-MATEX, which can be further improved using advance matrix solvers, such as [58].

tial integration scheme. We visualize the error distributions to show the advantages of using rational (R-MATEX) and invert (I-MATEX) Krylov subspace methods for matrix exponential and vector product (MEVP) over standard Krylov subspace method (MEXP). For the PDN simulation, our time integration scheme can perform adaptive time stepping without repeating matrix factorizations, which cannot be achieved by traditional methods using implicit numerical integration with fixed time-step scheme. Compared to the commonly adopted framework TR with fixed time step (TR-FTS), our single mode framework (R-MATEX) gains runtime speedup up to around 15×. We also show that the distributed MATEX framework (DR-MATEX) leverages the superposition property of linear system and decomposes the task based on the feature of input sources, so that we reduce chances of Krylov subspace generations for each node. We achieve runtime improvement up to 98×

 speedup.

We show the exponential integration with Krylov subspace methods maintains high order accuracy and flexible time stepping ability. The exponential integration framework was actually mentioned by the very early work in circuit simulation algorithms [29], but it had not attracted too much attention due to the high computational complexities of matrix exponential during that time. Nowadays, the progress of Krylov subspace methods provides efficient way to compute matrix exponential and vector product, so that we can utilize certain features of exponential integration, which are hardly obtained by traditional time integration schemes. Exponential integration can also serve as stable explicit schemes [59], [60] for general dynamical systems. It is a promising framework for the future circuit simulation algorithms and software. The opportunities of parallel and distributed computing with the cutting-edge multi-core and many-core hardware are also worth exploring for the further parallelism and runtime improvement.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we propose an efficient framework MATEX for accurate PDN time-domain simulation based on the exponen-

ACKNOWLEDGMENT

We thank Prof. Scott B. Baden, Prof. Mike Botchev, Dr. Quan Chen, Dr. Peng Du, Dr. Martin Gander, Prof. Nicholas

TABLE III

THE PERFORMANCE OF DR-MATEX (DISTRIBUTED R-MATEX). **Group #**: GROUP NUMBER OF THE TESTCASES. THIS NUMBER REPRESENTS THE TOTAL NUMBER OF SIMULATION SUB-TASKS FOR THE DESIGN; **Tran(s)**: RUNTIME OF TRANSIENT SIMULATION AFTER DC (SECONDS); **Total(s)**: RUNTIME OF OVERALL TRANSIENT SIMULATION (SECONDS); **Max. Df.(V)** AND **Avg. Df.(V)**: MAXIMUM AND AVERAGE DIFFERENCES COMPARED TO THE SOLUTIONS OF ALL OUTPUT NODES PROVIDED BY IBM POWER GRID BENCHMARKS. **SPDP_{tr}**: SPEEDUP OVER TR-FTS'S **Tran(s)** IN TABLE II; **SPDP_r**: SPEEDUP OVER R-MATEX'S **Tran(s)** IN TABLE II; **Peak m**: THE PEAK DIMENSION USED IN DR-MATEX FOR MEVP; **Mem. Ratio over TR-FTS**: THE PEAK MEMORY COMPARISON BETWEEN THE MAXIMUM MEMORY CONSUMPTION OF DR-MATEX OVER TR-FTS IN TABLE II.

Design	DR-MATEX					Speedups		Peak <i>m</i>	Mem. Ratio over TR-FTS
	Group #	Tran(s)	Total(s)	Max Df.(V)	Avg Df.(V)	SPDP _{tr}	SPDP _r		
ibmpg1t	100	1.4	1.9	5.3e-5	8.6e-6	4.0×	7.1×	6	1.9
ibmpg2t	100	8.9	11.4	4.6e-5	8.6e-6	4.5×	4.0×	7	1.9
ibmpg3t	100	91.7	129.9	9.6e-5	19.7e-6	2.9×	4.4×	6	1.5
ibmpg4t	15	52.3	112.2	9.9e-5	27.9e-6	8.8×	3.8×	8	1.4
ibmpg5t	100	148.4	178.9	9.0e-5	1.1e-6	3.2×	2.6×	7	1.5
ibmpg6t	100	189.9	234.2	3.4e-5	7.2e-6	3.8×	4.1×	7	1.5
ibmpg1t_new	100	2.4	2.8	5.3e-5	8.6e-6	21.8×	5.0×	6	1.9
ibmpg2t_new	100	5.6	7.0	4.6e-5	8.6e-6	61.6×	6.2×	7	1.9
ibmpg3t_new	100	103.0	140.9	9.8e-5	19.9e-6	25.6×	3.3×	7	1.5
ibmpg4t_new	15	51.5	108.4	9.9e-5	27.6e-6	98.0×	6.8×	8	1.4
ibmpg5t_new	100	185.6	227.8	9.9e-5	2.2e-6	27.3×	4.0×	7	1.5
ibmpg6t_new	100	274.8	317.7	3.4e-5	7.1e-6	25.5×	3.3×	7	1.5
Average	—	—	—	7.1e-5	12.3e-6	26×	5×	6.7	1.6

Higham, Prof. Marlis Hochbruck, Junkai Jiang, Dr. John Lof-feld, Dr. Jingwei Lu, Mulong Luo, Prof. Alexander Ostermann, Prof. Mayya Tokman and Chicheng Zhang, for the informative discussions. Hao Zhuang thanks the supports from UCSD's Powell Fellowship and Qualcomm FMA Fellowship. Last but not the least, we thank for all the insightful comments from the reviewers.

REFERENCES

- [1] D. Kouroussis and F. N. Najm, "A static pattern-independent technique for power grid voltage integrity verification," in *Proc. IEEE/ACM Design Autom. Conf.*, pp. 99–104, 2003.
- [2] S. R. Nassif and J. N. Kozhaya, "Fast power grid simulation," in *Proc. IEEE/ACM Design Autom. Conf.*, pp. 156–161, 2000.
- [3] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Proc. DATE*, pp. 1–6, 2007.
- [4] S. Lin, M. Nagata, K. Shimazake, K. Satoh, M. Sumita, H. Tsujikawa, and A. T. Yang, "Full-chip vectorless dynamic power integrity analysis and verification against 100uv/100ps-resolution measurement," in *Proc. IEEE CICC*, pp. 509–512, 2004.
- [5] S. Lin and N. Chang, "Challenges in power-ground integrity," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 651–654, 2001.
- [6] R. Zhang, B. H. Meyer, W. Huang, K. Skadron, and M. R. Stan, "Some limits of power delivery in the multicore era," *WEED*, 2012.
- [7] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *Proc. IEEE/ACM Intl Symp. Computer Architecture*, pp. 373–384, 2014.
- [8] K. Wang, B. H. Meyer, R. Zhang, K. Skadron, and M. R. Stan, "Walking pads: Fast power-supply pad-placement optimization," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf.*, vol. 20, p. 4, 2014.
- [9] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. Huang, C.-C. Teng, and C.-K. Cheng, "ePlace: Electrostatics based placement using Nesterov's method," in *Proc. IEEE/ACM Design Autom. Conf.*, 2014.
- [10] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 48–55, 2005.
- [11] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C. K. Cheng, "ePlace-MS: Electrostatics based placement for mixed-size circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 685–698, 2015.
- [12] L. Xiao, Z. Xiao, Z. Qian, Y. Jiang, T. Huang, H. Tian, and E. F. Y. Young, "Local clock skew minimization using blockage-aware mixed tree-mesh clock network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 458–462, 2010.
- [13] Y. Zhang and C. Chu, "GDRouter: Interleaved global routing and detailed routing for ultimate routability," in *Proc. IEEE/ACM Design Autom. Conf.*, pp. 597–602, 2012.
- [14] A. B. Kahng, S. Kang, H. Lee, I. L. Markov, and P. Thapar, "High-performance gate sizing with a signoff timer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 450–457, 2013.
- [15] C. Zhuo, G. Wilke, R. Chakraborty, A. Aydin, S. Chakravarty, and W.-K. Shih, "A silicon-validated methodology for power delivery modeling and simulation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 255–262, 2012.
- [16] Z. Zeng, X. Ye, Z. Feng, and P. Li, "Tradeoff analysis and optimization of power delivery networks with on-chip voltage regulation," in *Proc. IEEE/ACM Design Autom. Conf.*, pp. 831–836, 2010.
- [17] H. Zhuang, J. Lu, K. Samadi, Y. Du, and C. K. Cheng, "Performance-driven placement for design of rotation and right arithmetic shifters in monolithic 3D ICs," in *Proc. IEEE Intl. Conf. Communications, Circuits and Systems*, vol. 2, pp. 509–513, 2013.
- [18] S. K. Samal, K. Samadi, P. Kamal, Y. Du, and S. K. Lim, "Full chip impact study of power delivery network designs in monolithic 3D ICs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 565–572, 2014.
- [19] D. Xiang and K. Shen, "A thermal-driven test application scheme for pre-bond and post-bond testing of three-dimensional ICs," *ACM Journal on Emerging Technologies of Computing Systems*, vol. 10, no. 2, p. Art. no. 18, 2014.
- [20] H. Zhuang, X. Wang, I. Kang, J.-H. Lin, and C. K. Cheng, "Dynamic analysis of power delivery network with nonlinear components using matrix exponential method," in *IEEE Intl. Symp. EMC&SI*, 2015.
- [21] H. Zhuang, W. Yu, I. Kang, X. Wang, and C. K. Cheng, "An algorithmic framework for efficient large-scale circuit simulation using exponential integrators," in *Proc. IEEE/ACM Design Autom. Conf.*, 2015.
- [22] S. R. Nassif, "Power grid analysis benchmarks," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf.*, pp. 376–381, 2008.
- [23] Z. Ye, Z. Zhu, and J. R. Phillips, "Sparse implicit projection (SIP) for reduction of general many-terminal networks," in *Proc. Intl. Conf. Computer-Aided Design*, pp. 736–741, 2008.
- [24] Z. Li, R. Balasubramanian, F. Liu, and S. Nassif, "2012 tau power grid simulation contest: benchmark suite and results," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 643–646, 2012.
- [25] C. Zhuo, H. Gan, and W.-K. Shih, "Early-stage power grid design: Extraction, modeling and optimization," in *Proc. IEEE/ACM Design Autom. Conf.*, pp. 1–6, 2014.
- [26] H. Zhuang, W. Yu, G. Hu, Z. Liu, and Z. Ye, "Fast floating random walk algorithm for multi-dielectric capacitance extraction with numerical characterization of Green's functions," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf.*, pp. 377–382, 2012.
- [27] C. Zhang and W. Yu, "Efficient space management techniques for large-scale interconnect capacitance extraction with floating random walks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1633–1637, 2013.

- [28] W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu, "RWCAP: A floating random walk solver for 3-D capacitance extraction of very-large-scale integration interconnects," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 3, pp. 353–366, 2013.
- [29] L. O. Chua and P.-M. Lin, *Computer Aided Analysis of Electric Circuits: Algorithms and Computational Techniques*. Prentice-Hall, 1975.
- [30] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
- [31] T. A. Davis, *Direct Method for Sparse Linear Systems*. SIAM, 2006.
- [32] T. Yu and M. D.-F. Wong, "PGT_SOLVER: An efficient solver for power grid transient analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 647–652, 2012.
- [33] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "Powerrush: Efficient transient simulation for power grid analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 653–659, 2012.
- [34] X. Xiong and J. Wang, "Parallel forward and back substitution for efficient power grid simulation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 660–663, 2012.
- [35] L. Nagel, *SPICE2: A computer program to simulate semiconductor circuits*. Ph.D. dissertation, 1975.
- [36] S.-H. Weng, Q. Chen, and C. K. Cheng, "Time-domain analysis of large-scale circuits by matrix exponential method with adaptive control," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 8, pp. 1180–1193, 2012.
- [37] Y. Saad, "Analysis of some krylov subspace approximations to the matrix exponential operator," *SIAM J. Numer. Anal.*, vol. 29, no. 1, pp. 209–228, 1992.
- [38] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [39] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *CACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [40] D. Xiang, Y. Zhang, and Y. Pan, "Practical deadlock-free fault-tolerant routing in meshes based on the planar network fault model," *IEEE Trans. Computers*, vol. 58, no. 5, pp. 620–633, 2009.
- [41] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *HotCloud*, pp. 10–10, 2010.
- [42] Q. He, W. Au, A. Korobkov, and S. Venkateswaran, "Parallel power grid analysis using distributed direct linear solver," in *IEEE Intl. Symp. EMC*, pp. 866–871, 2014.
- [43] N. Gupte and J. Wang, "Secure power grid simulation on cloud," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 422–432, 2015.
- [44] J. Wang and X. Xiong, "Scalable power grid transient analysis via MOR-assisted time-domain simulations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2013.
- [45] X.-X. Liu, H. Wang, and S. X.-D. Tan, "Parallel power grid analysis using preconditioned gmres solver on CPU-GPU platforms," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 561–568, IEEE, 2013.
- [46] Z. Feng and P. Li, "Multigrid on GPU: tackling power grid analysis on parallel simt platforms," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 647–654, 2008.
- [47] H. Zhuang, S.-H. Weng, J.-H. Lin, and C. K. Cheng, "MATEX: A distributed framework of transient simulation of power distribution networks," in *Proc. IEEE/ACM Design Autom. Conf.*, 2014.
- [48] M. J. Gander and S. Guttel, "PARAEXP: A parallel integrator for linear initial-value problems," *SIAM Journal on Scientific Computing*, vol. 35, no. 2, pp. C123–C142, 2013.
- [49] S.-H. Weng, Q. Chen, N. Wong, and C. K. Cheng, "Circuit simulation via matrix exponential method for stiffness handling and parallel processing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 407–414, 2012.
- [50] J. van den Eshof and M. Hochbruck, "Preconditioning Lanczos approximations to the matrix exponential," *SIAM J. Sci. Comput.*, vol. 27, no. 4, pp. 1438–1457, 2006.
- [51] M. A. Botchev, "A short guide to exponential krylov subspace time integration for maxwell's equations," Dept. of Applied Mathematics, Univ. of Twente, 2012.
- [52] H. Zhuang, S.-H. Weng, and C. K. Cheng, "Power grid simulation using matrix exponential method with rational krylov subspaces," in *Proc. IEEE Intl. Conf. ASIC*, 2013.
- [53] Q. Chen, S.-H. Weng, and C. K. Cheng, "A practical regularization technique for modified nodal analysis in large-scale time-domain circuit simulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 1031–1040, 2012.
- [54] J. Wilkinson, "Kronecker's canonical form and the QZ algorithm," *Linear Algebra and its Applications*, vol. 28, pp. 285–303, 1979.
- [55] P. Li, "Parallel circuit simulation: A historical perspective and recent developments," *Foundations and Trends in Electronic Design Automation*, vol. 5, no. 4, pp. 211–318, 2012.
- [56] X. Ye, M. Zhao, R. Panda, P. Li, and J. Hu, "Accelerating clock mesh simulation using matrix-level macromodels and dynamic time step rounding," in *ISQED*, pp. 627–632, 2008.
- [57] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in *NSDI*, pp. 22–22, 2011.
- [58] X. Chen, Y. Wang, and H. Yang, "NICS LU: An adaptive sparse matrix solver for parallel circuit simulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 2, pp. 261–274, 2013.
- [59] M. Hochbruck and A. Ostermann, "Exponential integrators," *Acta Numerica*, vol. 19, pp. 209–286, 2010.
- [60] M. Hochbruck, C. Lubich, and H. Selhofer, "Exponential integrators for large systems of differential equations," *SIAM J. Sci. Comput.*, vol. 19, no. 5, pp. 1552–1574, 1998.



Hao Zhuang (S'11) is a Ph.D. candidate at the Department of Computer Science and Engineering, University of California, San Diego, CA, USA (UC San Diego). He received his C.Phil. degree in Computer Science from UC San Diego in June, 2015. His current research interests include numerical and optimization algorithms, with the applications in design automation, very large-scale integration systems, network and data analysis.

He has industrial experiences in several companies, including Synopsys, Inc., Mountain View, CA, USA, and Ansys, Inc., San Jose, CA, USA, where he worked on the large-scale circuit analysis and dynamic power network simulation algorithms within several products. He was one of the main software developers of RWCAP, a parallel program for VLSI capacitance extraction using floating random walk algorithm at Tsinghua University. At UC San Diego, he designed the numerical algorithms for large-scale circuit simulation using matrix exponentials, and optimization algorithms for electrostatics based VLSI global placement.

Mr. Zhuang was the recipient of the Charles Lee Powell Fellowship and the Qualcomm FMA Fellowship. He is a student member of IEEE, ACM and SIAM. He serves as a reviewer for IEEE Transactions on Computer Aided Design (TCAD), and an external reviewer for Design Automation Conference (DAC) and International Symposium on Physical Design (ISPD).



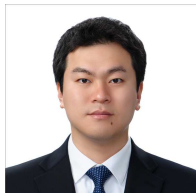
Wenjian Yu (S'01-M'04-SM'10) received the B.S. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 1999 and 2003, respectively.

In 2003, he joined Tsinghua University, where he is an Associate Professor with the Department of Computer Science and Technology. He was a Visiting Scholar with the Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA, twice during the period from 2005 to 2008. He serves as an associate editor of IEEE Transactions on Computer Aided Design since 2016. His current research interests include numerical modeling and simulation techniques for designing complex systems (like integrated circuit, touchscreen, cyber physical system, etc.), and the matrix computation algorithms for BIG DATA analysis. He has authored three books and over 130 papers in refereed journals and conferences.

Dr. Yu was a recipient of the Distinguished Ph.D. Award from Tsinghua University in 2003 and the Excellent Young Scholar Award from the National Science Foundation of China in 2014.



Shih-Hung Weng is currently with Facebook as Research Scientist and working on large scale distributed system for payments services. He received the B.S. and M.S. degrees in Computer Science from National Tsing Hua University, Hsinchu, Taiwan, in 2006 and 2008, and the Ph.D. degree from University of California San Diego, La Jolla, U.S.A in 2013 under the supervision of Dr. Chung-Kuan Cheng. His thesis focused on parallel circuit simulation for power-grid noise analysis in very large scale integration (VLSI) system.



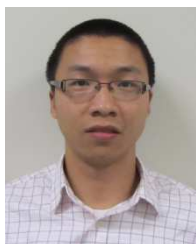
Ilgweon Kang (S'12) received the B.S. and M.S. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree at University of California at San Diego (UCSD). He was with the Research and Development Division, SK Hynix, Icheon, Korea, from 2008 to 2012, where he was involved in development of 3D stacked DRAM with (Through Silicon-Via) TSV technology.

His current research interests include 3D IC design and optimization, CAD, and design for testability



Jeng-Hau Lin (S'15) is a Ph.D student at Department of Computer Science and Engineering, University of California, San Diego, CA, USA. He received his B.S degree in Electrical Engineering in 2005 and M.S. degree in Communication Engineering in 2007 from National Taiwan University.

His research interests includes time-frequency analyses on signal integrity and the consensus of multi-agent system.



Xiang Zhang (M'13) received the B.Eng. in Information Engineering from Shanghai Jiao Tong University (SJTU), China and M.S. in Electrical and Computer Engineering from University of Arizona, Tucson, AZ, in 2010.

From 2011 to 2014, he was a Senior Hardware Engineer with Qualcomm Technologies, Inc. In 2014, he joined Apple Inc as an iPhone Hardware System Engineer. Since 2012, he has been working towards the Ph.D. degree in Department of Electrical and Computer Engineering at University of California,

San Diego, CA, USA.

His current research interests include power distribution network design and optimization, circuit simulation and system-on-chip design.



Ryan Coutts received a B.S. in electrical engineering from the University of California, Riverside in 2005. He then received his M.S. in electrical engineering from Stanford University in 2006. After receiving his M.S. degree, he worked as a signal and power integrity engineer at NVIDIA Inc. specializing in FSB, DDR and power integrity. He is currently working at Qualcomm Inc. where he works in low power design methodologies for mobile covering the range from power integrity, thermal optimization and on-chip timing. Mr. Coutts has filed 13 patents

related to his innovation as well and is currently pursuing a Ph.D at the University of California, San Diego.



Chung-Kuan Cheng (S'82-M'84-SM'95-F'00) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, and the Ph.D. degree in electrical engineering and computer sciences from University of California, Berkeley in 1984.

From 1984 to 1986 he was a senior CAD engineer at Advanced Micro Devices Inc. In 1986, he joined the University of California, San Diego, where he is a Distinguished Professor in the Computer Science and Engineering Department, an Adjunct Professor in the Electrical and Computer Engineering Department. He served as a principal engineer at Mentor Graphics in 1999. He was an associate editor of IEEE Transactions on Computer Aided Design for 1994-2003. He is a recipient of the best paper awards, IEEE Trans. on Computer-Aided Design in 1997, and in 2002, the NCR excellence in teaching award, School of Engineering, UCSD in 1991, IBM Faculty Awards in 2004, 2006, and 2007, the Distinguished Faculty Certificate of Achievement, UJIMA Network, UCSD in 2013. He is appointed as an Honorary Guest Professor of Tsinghua University 2002-2008, and a Visiting Professor of National Taiwan University 2011, and 2015.

His research interests include medical modeling and analysis, network optimization and design automation on microelectronic circuits.